Markku Wiik

Integrating 3d into the 2d game character work flow

Thesis
Kajaani University of Applied Sciences
Business
Business Information Systems
20.3.2013

KAJAANIN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

THESIS ABSTRACT

School Business	Degree Programme Business Information Systems
Author(s) Markku Wiik	
Title Integrating 3d into the 2d game character workflow	
Optional Professional Studies	Instructor(s) Nick Sweetman
	Commissioned by
Date Spring 2013	Total Number of Pages and Appendices 26 + 1

This thesis concentrates on how to design game characters but also tells some of the needed steps to make them in 3d software. The design chapters tells what points should be thought when designing a character and how to proceed from an idea to a final character concept. The next chapters briefly describes modeling, texturing, animating and rendering steps when making a 3d game character.

The practical part was done as a part of small game project where a group of students started to develop a 2d platformer game. The game was not finished at the end of the project. This thesis was written by the projects graphical artist and includes the game's character designs and a brief chapter about the game's environment design.

Language of Thesis	English
Keywords	game graphics 2d 3d
Deposited at	☒ Electronic library Theseus ☐ Library of Kajaani University of Applied Sciences

SISÄLLYS

LIST OF APPENDICES

SYMBOL LIST

Color Hue – Different colors, for example green, red and blue.

Color Chroma – Measures colors deepness. Low chroma is near gray. Related to Saturation.

Color Saturation – How vivid a color is. Low saturation is near gray.

Color Value – The lightness of a color. Color with low value is near black.

Enveloping – Adding bones to a 3D-mesh and adjusting how the bones deform the mesh.

HUD – Heads-Up Display. Information graphics shown on the game screen for example health bar and score display.

Instanced copy – Creates a completely interchangeable clone of the original.

Mesh – 3D-object.

Polygon – A plane formed by three or more vertices.

Power-up – Collectible game object.

Rigging – Building controls for bones. Animating is done by moving the rig.

Sprite – Two-dimensional image.

Sprite sheet – Collection of multiple sprites in one image.

Texturing – Painting textures with aid of an UV-map.

Tile – An image used to build terrains.

UI – User Interface. Game menu or similar.

UV-mapping – Unfolding 3D-object to 2D image. UV-map is used for texturing.

Vertex – Points between edges in a 3D-object.

1 INTRODUCTION

This thesis was done as a part of team developing a 2d platformer game. In the game player controls a robot and tries to travel as far as possible. The mock-up image (Picture 1.) shows the player robot with a green box head and four different enemy robots. The team was formed by four members: a game designer, two programmers and one graphics artist. Every team member was meant to write their thesis for the game project.

Picture 1. Game screen mockup

At first this thesis was meant to give insight on how a graphics artist worked in a team of students making a game, but this was not suitable subject for thesis. Later the subject was changed to cover how 3d was used to produce 2d game graphics.

The team used three months developing the game, but at the end the game was still unfinished and unplayable.

Software used for the graphics: Adobe Photoshop, Corel Painter and Autodesk Softimage.

2 CHARACTERS

Characters are a major part in games, they bring the game to life. There are characters the player can control: the heroes. We might have feelings towards the main characters, care, fear. Then there are NPCs (Non Player Characters) who have wide variety of roles. NPCs can help keeping the story going or they might be obstacles or goals. Characters can pull us into the game they might provide humor and add color to the game. (Angel 2008.)

A thing that separates game characters and characters in films is that a player can control his character's actions in a game and the players decisions affect how the story goes. (Angel 2008.)

2.1 Character Design

In a game every character should have a purpose, and the character needs to be designed to fill that purpose. Every part in the character also should have a meaning. Even the most creative character design is wasted if it does not fit in the game. (Seegmiller 2007, 1,6.) An important thing to know is what the character needs to do in the game. "A guy with a gun" is not enough information. Usually the basics should have at least age, height, sex and hair/eye/skin color. A brief history or biography of the character will help to understand the character's personality which may answer the questions what the character might wear and why they wear it.

If the character design is based on someone's ideas, the artist is supposed to satisfy their vision of that character. Creating a successful character based on another person's work needs a lot communication between the designer and the artist. If the designer keeps changing his ideas, it will make the artist's job harder. (Ward 2005, 5; Steed 2005, 10.) Everyone views the world somewhat differently. People will have different mental images formed when they hear identical words or even view identical images. It is very important to make sure both parties understand exactly what the artist is visually and technically asked to do. An artist should ask lots of questions to make sure the task is clear and both have the same idea. (Seegmiller 2007, 14-15.)

List for points to think when designing a character:

- Target Audience – Whom is the game for? The game's design and style should appeal to right age group. (Ward 2005, 3.)

- Color – Colors are usually associated with certain emotions and personalities. Evil and rebellious characters usually wear dark colors. Good and pure persons are more likely to have brighter color schemes. (Ward 2005, 17.)
- Player Identification – Players should be able to identify with the game hero. (Ward 2005, 3.)

- Branding – Main characters will usually be used in advertisements to promote the game. The main characters should be original and memorable. (Ward 2005, 3.)

- Technical Considerations – Character designs will be limited by the gaming platform. For a 3D game limitations usually are: polygon count, texture page size, animation data and joint limits. (Ward 2005, 3.)

- The Game World – The characters should fit to their world. Competing styles will make the characters look out of place and the player may lose their connection with the game hero. (Ward 2005, 3.)

- How Will the Character Be Used – Is the character only a supporting object, a mid-ground prop or will the character be the center of interest. These will affect the needs of detail. (Seegmiller 2007, 8.)

- Will the Character Be Animated – With a stationary character there is no need to think how it's joints would be placed. (Seegmiller 2007, 9.)

- Viewing Angle – Will the character be viewed from more than one angle. Proportional distortions are often needed in games which have top-down or isometric view. Usually in these cases head and shoulders are exaggerated and limbs might be thickened. (Ward 2005, 13; Seegmiller 2007, 9.)

- Contrast Among Characters – When designing multiple characters, it is important to have contrast between them. There should not be two too similar characters. The characters should have distinguishable silhouettes. (Ward 2005, 17.)

- Asymmetry – No one has completely symmetrical features. A good way to make characters more interesting, is to add variation with asymmetry. (Ward 2005, 17.)

Coming up with an idea of a character can be hard and the idea is only the first step. Seegmiller says that erasing while drawing will kill the creative flow and should be avoided. To help developing a basic idea into more specified visual image Seegmiller gives plenty tips including the following: caricatures, blotter images, doodling. Drawing caricatures may help to find a stalled character's essence again. Blotter images help to brew new ideas from random shapes. With blotter pictures Seegmiller means images such as Rorschach test uses as seen in Picture 2. Doodling and scribbling is a great way to form ideas quickly. (Seegmiller 2007, 24-32.)

Picture 2. Rorschach test image

When independently developing a game without any licenses, all the designing will be done in house. Approval times will be shorter and there will be more creative freedom, but it will take more work when starting a character from nothing. Using licensed characters can be restricting, the characters must be true to its original style and design. Accomplishing staying true to the original design will involve getting approval from the character's original designers or license holders which can lengthen the designing process. (Ward 2005, 2.) Seegmiller sees designing characters as problem solving and lists five steps to take. (Seegmiller 2007, 17.)

1. Identify the expectations for the character. Define the problem.

2. Analyze and simplify the problem into smaller problems. Generate ideas to solve the smaller problems. Combine the answers to solve the original problem.

3. Choose the best ideas for a character.

4. Draw the character.

5. Evaluate the resulting drawings and have others to look the results too.

Showing the art to others is important because the artists can become blind to the bigger picture and errors he has made. (Ward 2008, 18.) Ward suggests drawing rough sketches and playing with ideas for a few days. Cleaner version can be created when feeling confident and comfortable with an idea. This cleaner version rarely is the final design. (Ward 2005, 20.)

2.2 3d modeling

One commonly used method for modeling is to start the character from several primitive shapes, for example cylinders and cubes, and shapes them to fit the guide images. Then Ward welds the shapes together into one mesh and continues adding details and needed geometry. (Ward 2005, 53-132.) Sims has a different method, he extrudes selected polygons which will form desired part of a body, for example a tail or legs. (Sims & Isner, 2004, 24-41.) Usually it is enough to model just one half of a character. The half is then instanced so the modeler can still see the character as a whole. After the one half is ready it is mirrored to the other side and the modeler can continue editing the whole mesh. (Ward 2005, 29,37.)

2.3 Texturing

During the texturing phase the character gets its color and texture. (Sims & Isner, 2004, 45.)

Before being able to draw textures for a character it must have a material and an UV-map applied. UV-map is a 2D-presentation of a 3D-model. It is made of UV-points, which are 2D coordinates that correspond to specific pixels on a texture. According to ward, the most efficient method to texture a character is to have it divided into easily mappable shapes. Easily mappable shapes mean areas that resemble UV-projection shapes, usually cylinders or planar. (Ward 2005, 323-339.)

The UV-map will need editing after it has been generated. Using a checkered texture will make it easier so spot problem areas. On the problem areas the checker pattern will be distorted or stretched. (Ward 2005, 340.)

The UV-map will be exported as an image file. Textures will be painted in a 2D-painting software using the UV-map as a guide. Ward starts with first painting flat base colors for each area. With the base colors painted it is good idea to check how the seams are aligned in

3D-software before starting to paint in the details for the texture. Ward uses separate layers for base colors, shades and highlights. A game can have several texture pages for one model's area. The different texture pages are diffusion: color and shading, alpha: transparency, specular: bouncing light, bump maps or normal: add surface details. (Ward 2005, 391-441.)

Sometimes it bay be easier to have an image and then adjust the UV-map so it looks good on the character.

2.4 Animation

Before starting to animate, the animator should know everything the character needs to do and how the character is going to perform those actions. (Ward 2005, 710.)

In animations timing has to be correct, movements can be too slow or too fast which will result the whole animation looking awkward. In animation emphasis means exaggeration and dynamism to make movements more readable by the audience. Emphasizing too many actions at the same time will confuse the viewers. Secondary motion includes movements that leads to or is a result of a primary motion. An example of secondary motion is the flopping ears of a rabbit while it jumps. (McKinley 2006, 14-17.)

Anticipation is a movement that hints to upcoming main action. A little motion that prepares the audience for a major action, for example a human bends down a little to build momentum for a jump. (McKinley 2006, 17; Thomas & Johnston 1995 47-69.)

Follow through is a movement that is a result of a primary movement. When a character stops moving some parts still continue moving to catch up the main body's movement. For example long hair or a long coat. Nothing stops all at once. (Thomas & Johnston 1995 47-69.)

Overlapping action brings realism to animations. For example a gun holster hanging from a running cowboy's belt. Or when a character changes a direction while walking, the character's long coat will continue to move in the original direction before catching up with the main body. (McKinley 2006, 18; Thomas & Johnston 1995 47-69.)

The first step in this phase is creating the hierarchy of bones for the skeleton. The second step is enveloping the model to the skeleton. In the last step a rig is created and character will be ready for animating. (Softimage|XSI Character Animation 2006, 18.)

Bones in Softimage are usually referred as chains. Chains can be 2D or 3D chains, main difference being movement restrictions. 2D chains only rotate on one axis, like an elbow or a knee. 3D chains can rotate freely on any axis, like a dog's tail. A chain has a preferred angle, which determines its direction of movement when bending. Chains should be drawn little bent, so their preferred angle will be set right. (Softimage|XSI Character Animation 2006, 37-38, 42.) Chains can be hard to select when they are inside a model. To make make selecting easier can their shape be changed or alternatively shadow icons can be used. (Softimage|XSI Character Animation 2006, 49-54.)

A skeleton is used to pose or deform a model. An envelope is an object which moves and deforms following their deformers. Usually this means the actual mesh being deformed using a skeleton. Every single point in the character mesh is assigned to at least one deformer. Weights determine how strongly a point is affected by which deformers. Deformers can be assigned in two ways, based on distance or normals. After assigning the deformers the weight distribution usually has to be modified manually if there is more than one deformer involved. Weight can be done multiple ways, locally reassigning will result the selected area being fully influenced by single deformer. Modifying weights in a way that allows distributing the weights between several deformers is done either by painting or using weight editor. Painting offers more visual feedback while the weight editor allows modifying weight assignments numerically. (Softimage|XSI Character Animation 2006, 77, 85, 97-107.)

Techniques for enveloping and modeling will vary for each animator and modeler. A useful tip is to avoid star junctions near deforming areas on a mesh. Star junctions may cause problems when deformed. Weights can be mirrored with symmetrical models. X-ray mode seeing through objects and makes deformers visible while seeing the models surface. (Softimage|XSI Character Animation 2006, 81, 111, 120.)

The skeleton will determine how the model will move when animated. The term for assigning a skeleton for a model varies depending what software is used. With Softimage the process is called enveloping. For simple characters there may be no need to create a rig. Characters with complicated movements and many bones will be easier to animate with a rig. (Softimage|XSI Character Animation 2006, 14-15.) A rig is a term used when referring to the control objects of a character. The control objects are used to move the character

indirectly. The control objects move only bones inside the character and the bones deform the mesh. A character can be animated using only bones, but a rig will make it a lot more intuitive and easier. (Ward 2005, 487-590.)

After completing the rig, it is good to set a neutral pose for the character. Neutral pose sets translation, rotation and translation values to zero and scaling values to one. This helps resetting the character to its default neutral pose, sometimes referred as zeroing the character. It also simplifies animation curves and key values when base value is 0 and not 3,74589. (Softimage|XSI Character Animation 2006, 69-72, 175-176.)

Usually animation is first done at low level. Animating at low level means changing an object's values in its parameters. Different ways to do low level animation are: keyframing, path animation, constraints, linked parameters, expressions and scripted operators. Keyframing is storing the values of selected parameters. A Character is posed in keyframes and computer will fill the keys between by blending the previous and the next keyframe. The animation can then be edited using function curves which are graphic presentations of the animation. The function curves show the interpolation between keyframes. The Dopesheet can also be used to edit editing and viewing animation keyframes, especially for moving and copying keyframes. (Softimage|XSI Animation 2006, 24; Sims & Isner, 2004, 116, 124-133.)

Low level animation can be stored to action sources and used for high level animation in the Animation Mixer. (Sims & Isner, 2004, 133; Softimage|XSI Nonlinear Animation 2006, 24.)

The animation mixer is used to work with animation in a way that is nonlinear and non destructive. The animation mixer is not affected by the main time line and all animation work done with animation mixer does not destroy the original animation data. The original animation data is still available inside the animation source and can be edited if necessary. Animation sources are loaded in the animation mixer as animation clips which are instances of the original sources. In the animation mixed clips are represented by boxes on animation tracks. The clips can then be edited independently without affecting the original source or other instances of the same clip. Modifying the original source will affect all the clips that are instanced from it. (Softimage|XSI Animation 2006, 24; Softimage|XSI Nonlinear Animation 2006, 14-16, 24, 34, 66; Sims & Isner, 2004, 133-135.)

Transitions are used for smoothing animation between two or more clips. Transitions work with clips on the same track and with clips that are not one the same track. If the clips overlap the interpolation is a weighted blend between the animation values at each frame. If

the clips do not overlap, the last frame of the first clip and the first frame of the second clip are used for the interpolation. Clips that overlap can also be mixed together by adjusting the weights of clips. The clips weight slider determines how much of an influence each clip has in the resulting animation. (Softimage|XSI Nonlinear Animation 2006, 66-67, 82.) Clip effects can be used to add animation to a clip without affecting the source. (Softimage|XSI Nonlinear Animation 2006, 94.)

2.5 Rendering

Render passes are usually used to divide a scene into several layers. Each layer can contain different scene elements, like shadows, lighting or just colors. Different passes can be composed in 2D to create a complete image. Having different elements on own layers allows more accurate editing and re-rendering just a part of a scene. (Softimage|XSI Rendering 2006, 23-24.)

A partition is a division of a pass, similar to groups, it can contain can only contain lights or geometric objects. A pass can have as many partitions as needed. A partition is used to override properties for each object it contains, for example changing a shader. (Softimage|XSI Rendering 2006, 25-26.)

2.6 Sprite Sheets

The game engine uses tile sheets for animations. A tile sheet has all the needed frames tiled into one image. The frames are usually arranged in consecutive order. (8-Bit-Rocket, 2008.) In this project the rendered frames were painted over in image editing software and arranged into sprite sheets. Each characters had its own sprite sheets and an animation data file which defined correct frame size and frame count for each animation sequence.

3 THE GAME

The main character in the game is a robot that is constructed from changeable parts for its arms, legs, head and torso. During the game each part can be upgraded to a part with different properties, for example the arms can be fitted with projectile weapons instead of blades. The levels for the game are generated procedurally and was meant to have destructible terrain. During the levels the player will meet enemy robots with different behaviors, for example flying or jumping robots. The player needs to adjust his actions according to his equipment and enemy type to get past the enemies. The score in the game is determined by the distance the player can travel before dying.

The final game has only 2d graphics, but during the work process some characters were modeled in 3d. The working method was to first draw 2d concept art and then model the characters in 3d. The 3d models were then animated and rendered to 2d images. The final sprite sheets were painted over the rendered images. Some characters were simple enough and the 3d phase was skipped, and the frame sheets were painted straight based on the sketches. Only one character, Flybot, had textures made for the 3d-model because the bot was shown from more than one viewing angle during its animation.

Most of the characters had simple models and simple animations and could be animated by directly controlling their bones. Only the player character had a rig built to control its bones.

Toon shader was used for rendering to achieve an effect (cel shading and ink lines) which gave better details for painting 2d sprites over the rendered images. The player character had multiple parts which needed to be rendered separately. To make this easier render partitions were used.

The level backgrounds had three layers with different scrolling speeds to emphasize the illusion of depth (parallax scrolling). Terrain for the levels was constructed using two types of tiles: desert and trash blocks.

3.1 The Game's Character Designs

Designing the enemy characters started by drawing a few thumbnail silhouettes for each character based on the team's designer's descriptions. The most promising silhouette was

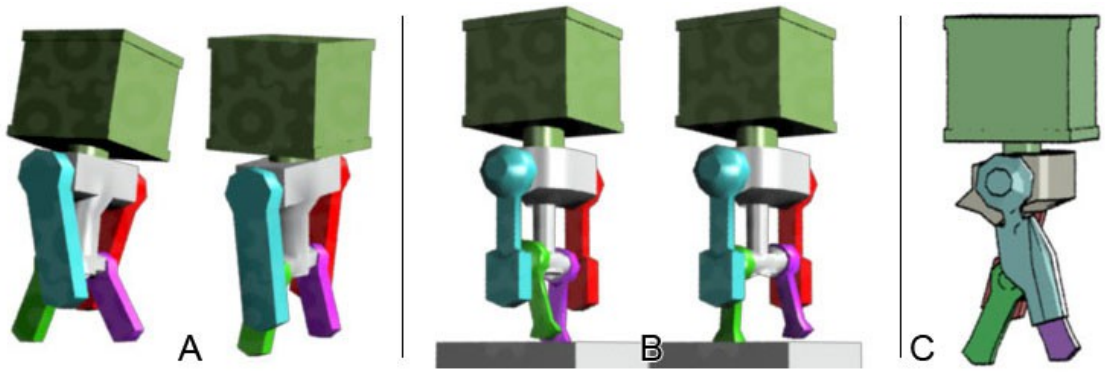
then scaled up and adjusted to look light gray. Next a few gray scale variations were drawn based on the light gray base images. One of these variations was then chosen and refined.

Robo

The first character that was drawn was the main character: Robo. The main character was going to be built from pieces. It would have several different heads, arms, eyes, mouths, legs and torso parts. Building from pieces made the graphics artist's work more complicated. The game designer had wrote descriptions for each part. The artist used a lot time thinking how the pieces should be done. Possibility to have different arms on each side made mirroring the character in game more complicated, as the parts would need to be rearranged depending which side Robo was facing.

The character was simple enough to be modeled without drawing detailed 2d images. Robo was the only character in the game that had some kind of rig done for animating. In the first walk cycle Robo was wobbling from side to side and each parts' viewing angle changed during the animation. Variations in viewing angle would have required a lot more work when painting the final sprite sheets and also shooting with a gun arm would be complicated because the hand going up and down and bullets would have been shot at different altitudes. The second version had moving hips which stabilized the upper torso and the arms. To make things easier the walk cycle was simplified to one where Robo's viewing angle stayed the same during each frame (Picture 3.).

Robo had many parts and all of them would have to be rendered separately. To make rendering a bit easier, render-pass partitions were used. Separate render-passes were created for each object that needed to be rendered alone. Only the required object would be visible in its own partition and unneeded objects would be hidden during rendering. This method removed the need to manually set objects visible or hidden each time a different object was rendered.



Picture 3. Different walk animations for Robo. A: full torso movement, B: Upper torso stabilized, C: No sideways motion

The Robo can have different arms on each side, which complicated mirroring the character's sprites because the parts would need to be rearranged. Shooting and jumping animations had more frames earlier but those were discarded. There was a possibility that Robo's arm rotations would be done in code and drawing the rotations was unnecessary. Robo was not textured at all. Coloring would be done by painting over the rendered images.

Robo had two head variations a green cube and a red sphere. Only one torso made from metal bars and a metal box. Both arms had three possible weapons: a basic gun, a close combat blade and a coil gun which shoots electricity. For legs Robo had two possible sets, normal sticks or a wheel (Picture 4.).

Picture 4. Robo's sprite sheet. Two heads, cube and sphere. Metallic torso. Three different weapons for each arm: coil gun, blade, "pea shooter". Leg variations: wheel or sticks. Three frames for a smoke effect.

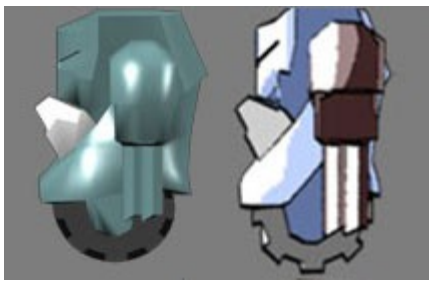
Guardbot

Guardbot was the first enemy bot that was designed. The game designer's idea for the GuardBot was a robot moving on a wheel and guarding the law. Guardbot has angular shapes which give an impression of roughness. Another design point is the bluish color theme with a sheriff star, which represents law enforcement and authority. In early sketches Guardbot had a hat that gave it a wild west attitude: He is the law. Reason for the hat's removal was that it was not a built part of the robot and not all Guardbots could have the same kind of hats. Another design point was size and placement of the gun arms. Original

position was at head height and covered the robots head when it was shooting. One method to fix this was shorten the weapon, but this made the weapon look too low powered. The other method was lowering the weapon under the head. The weapon looked the same, but left a disturbing empty area at its original position. Different versions can be seen in Picture 5.

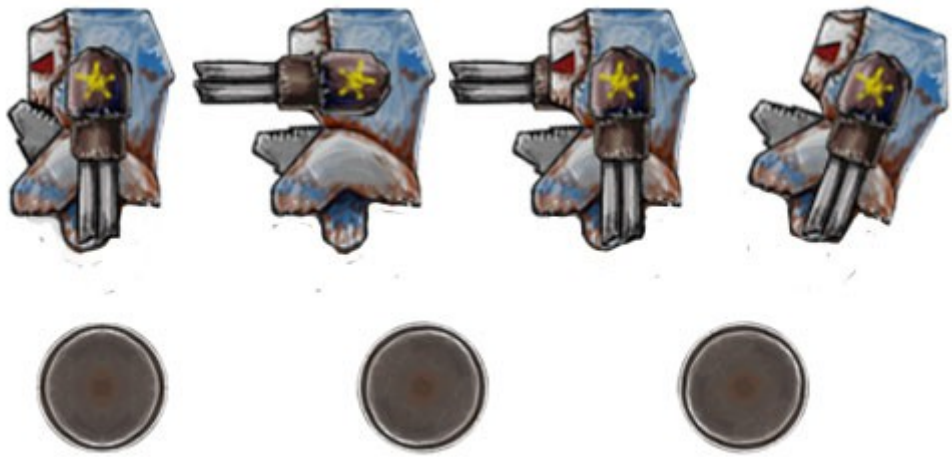
Picture 5. Guardbot gun placement variations

At first the rendering was done using the default phong shader but was changed to toon material shader for object colors and toon lens shader applied to camera for producing ink lines. Using toon effects helped distinguishing different shapes and made it easier to paint final sprite sheets. Using the toon lens shader forced to render using perspective camera instead of orthographic camera. One way to try correcting this was to adjust the fov-angle (Picture 6.).



Picture 6. Difference between phong and toon shader

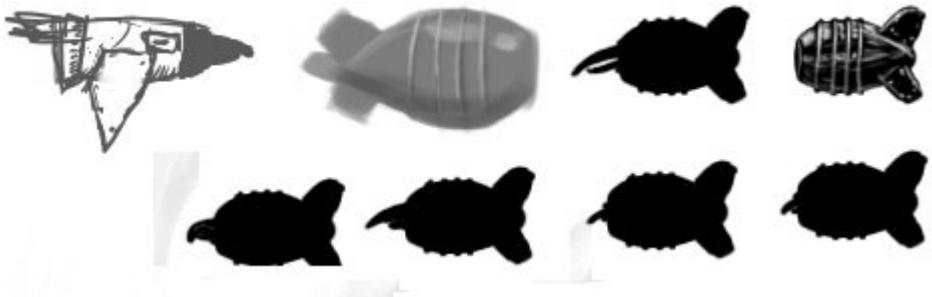
Guardbot had only four animations: moving, shooting with right arm, shooting with left arm and being hit. The guardbot's wheel was detached from its torso in the frame sheet, allowing to animate it independently (Picture 7).



Picture 7. Guardbot's frame sheet

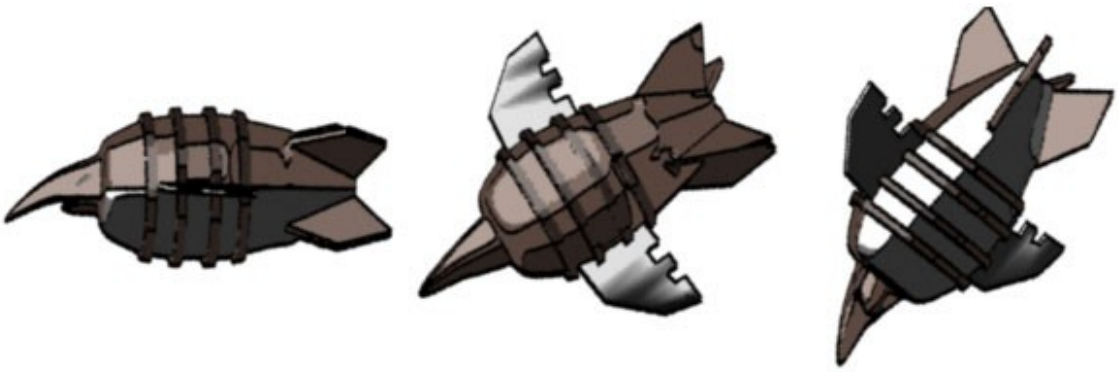
Flybot

Flybot was supposed to resemble an atomic bomb. The artist drew thumbnail images and experimented with different beak shapes (Picture 8.).

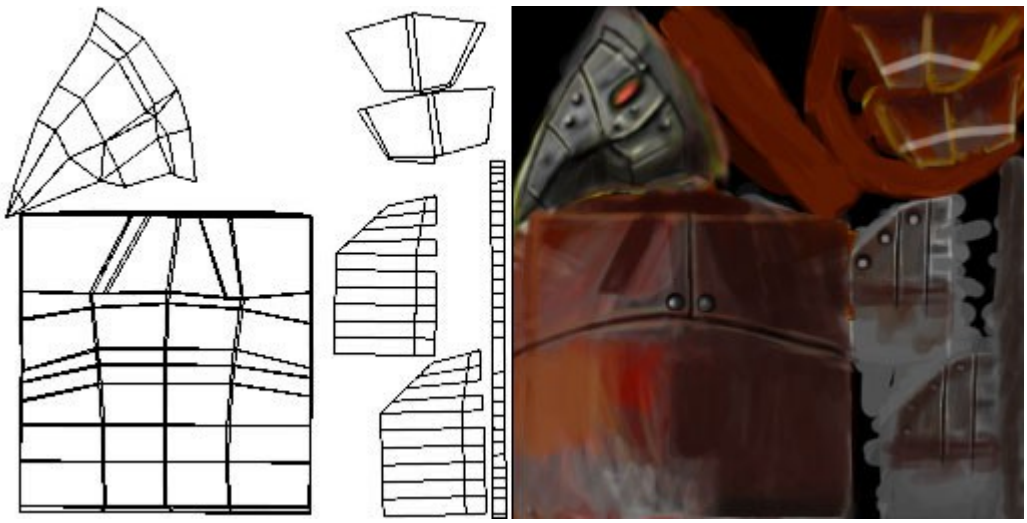


Picture 8. Different beak shapes for flybot

For FlyBot it was necessary to do detailed textures because it would rotate around its axis during the attack animation (Picture 9.). Painting textures for the Flybot model helped over painting the rendered animation frames. Warning colours were chosen for the colour scheme. The first version used yellow and orange, which made the bot look more like a bee so the color scheme was changed to red. The second texture version was also optimized by overlapping both halves (Picture 10.).

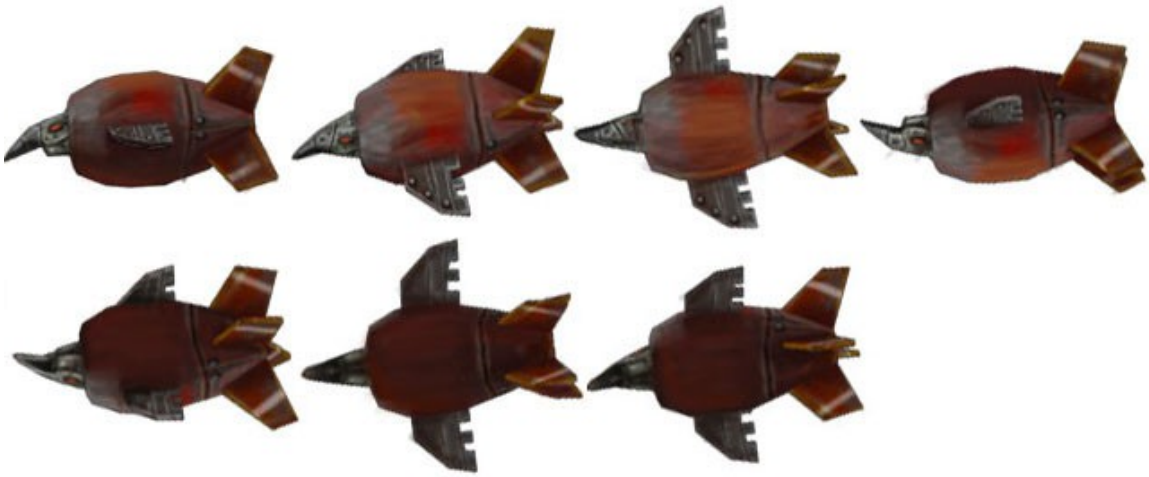


Picture 9. Flybot's rotation



Picture 10. Flybot's UV map and texture

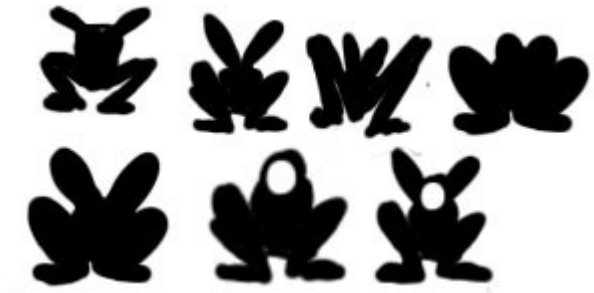
There was some over painting done with the rendered images for the final sprite sheet (Picture 11.), but painting the detailed texture saved time.



Picture 11. Flybot's frame sheet

Frobot

Frobot is built for jumping around like a frog or a rabbit, so frobot was designed to have long legs. Influence from frogs and rabbits can be seen in early silhouette sketches in Picture 12. The legs were a bit complicated to design. For reference the artist searched images of hydraulic cylinders. The reference images helped to design the mechanism for the legs (Picture 13.).



Picture 12. Frobot silhouettes

Picture 13. Frobot leg mechanism

Modeling frobot in 3d was seen unnecessary and it was also easier to do animations only in 2d. At the end frobot had a jumping animation made from two frames (Picture 14.).

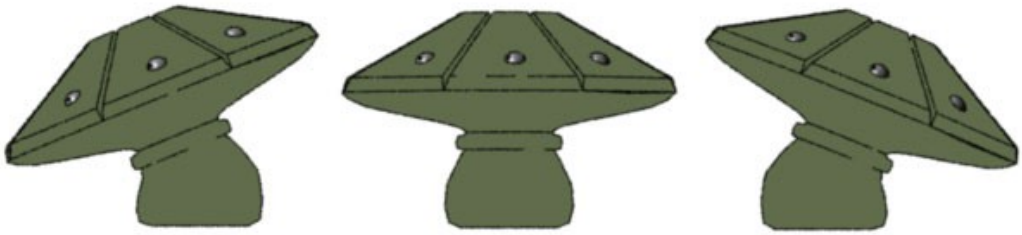
Picture 14. Frame sheet for frobot

Shroombot

Shroombot is a robot resembling a mushroom. It stands still and does not move. The artist drew one thumbnail silhouette and one small structure sketch. The shape for shroombot was simple and no further sketches were drew. For animation the idea was that shroombot would shake its top from side to side, then return standing straight and launch seeds around from its top (Picture 15.).

Picture 15. Shroombot shooting animation design

The artist did make a 3D model for shroombot but eventually it was not used. The 3D model had the shooting motion animated, but it did not look right and doing the animation only in 2d gave the possibility to add more distortion to the figure. The animation done in 3d can be seen in Picture 16.



Picture 16. Shroombot's animation done in 3d

In the 2d version, another details were added to shroombot: three eyes and a glowing effect. When the player is not near, shroombot stays still sometimes glowing and occasionally opening its eyes. If the player is near enough, shroombot closes its eyes and shakes a couple times and shoots some spores in the air. The final sprite sheet (Picture 17.) had three animations: glowing effect, opening and closing eyes and the shooting animation. The eyes were painted as separated frames so they could be open and close independently from the glowing effect. In the shooting animation the eyes would always be closed so they were combined with the shooting animation frames.

Illustration 17: Shroombot's sprite sheet

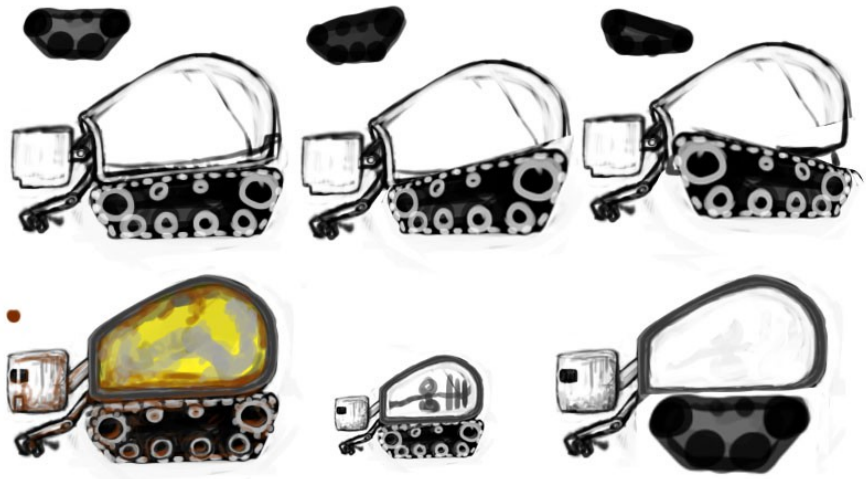
Turtlebot

Turtlebot's concept was a peaceful robot which would retaliate with a nuclear missile to any threat. Initial sketches were done at the beginning during the first meetings. A placeholder image was drawn for the first mockup shots.

The artist used the usual method and drew thumbnail images experimenting with different shapes and how the launching mechanism for the missile would work (Picture 18.). The artist also tried different track variations and colors (Picture 19.).

Turtlebot never had its sprite sheet done, at the end there was just one unfinished colored concept image (Picture 20.).

Picture 18. Turtlebot silhouette designs



Picture 19. Turtlebot track variations

Illustration 20: Turtlebot concept image

Engineer Sarai

Sarai is a female engineer and has built most of the robots. She is a very important character for the game's story but has no effect in game play. Sarai was planned only to be seen during the game's intro animation and at the game's end. She was designed to be wearing a white doctor's jacket and a bandanna covering a part of her head. There was only some sketches for her, no final design.

3.2 Environment

The game's environment was supposed to have destructible tiles as walkable platforms and layered scrolling images as background scenery.

The desert level was planned to have ground tiles and trash blocks on top of them. At first the ground tiles were 16*16 pixels in size. Using these small tiles to generate terrain resulted in somewhat too straight lines in the ground. The artist was not satisfied with the look of these small tiles and sketched an mock-up image where the tiles would be larger. The tiles were increased to 32*32 pixels and the artist followed Borzel's tutorial for drawing the tiles.

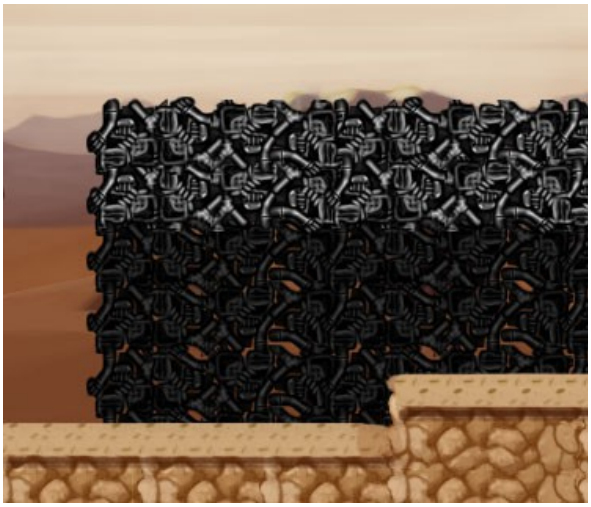
Tiles need to have connectable borders to make the connections seamless. The borders also include corners. Borzel starts drawing her tile sets from upper left corner tile. She draws the tile and then copies its borders for surrounding tiles creating seamless connection. Her advice is to never ever start with the center tile because it will be impossible to fine-tune its four borders with the outer tiles. Alternative view from a comment on her blog states that

the center tile is the most important tile of the whole set since it gets repeated often and it is not impossible to draw it first. (Borzel 2008).

The artist added a clear walkable path on the top tiles to make it easier for the player to distinguish where he is able to walk (Picture 14.).

Picture 21. Ground tiles without and with walkable path tiles

Another tile type for the levels was trash blocks. The trash blocks were not implemented in the game yet. The trash blocks were above the ground level and they were destructible. The first idea for the blocks was to use different mechanical objects but later the trash blocks were changed to piles of metallic junk. The trash tiles had graphics that was 50*50 pixels in size but would be tiled as 32*32 size tiles causing parts of them to overlap. This overlap was designed to enhance the "pile of junk" effect. The trash blocks had two color variations: light blocks and dark blocks. The light trash blocks were designed to be walkable platforms, while the player would be able to walk through the dark trash blocks which were meant to eliminate the effect of hovering blocks light trash blocks. Picture 15 shows how the trash blocks would have looked in the game.



Picture 22. Trash blocks

The background was made of different layers and used parallax scrolling. One layer had the sky and big sand dunes far in the horizon. Additionally there was three object levels which had sand dunes and random objects like an ankh, a robot's head, a crash landed flybot and piles of junk. Objects further away had their size, saturation and lightness adjusted to strengthen the illusion of distance (Picture 16.).

Picture 23. Background scenery and objects

4 EVALUATION

The original plan was to have a simple playable 2d platformer game with procedurally generated levels. During the design process other possible features were added to the list, like destructible terrain and an intro movie. At some point there was an almost playable version where the player could move around, jump and shoot in a procedurally generated level which also included three enemy robots: flybot, shroombot and frobot. The enemies were not fully functional. Later there were changes in the game engine which broke the game's update speed, moving seemed to happen one millimeter per second. This made it impossible to test the graphics in game.

At first the artist had a schedule, but at some point it become unnecessary. This might have been because there was no actual need to do any specific graphics, even almost finished graphics were not in the game yet.

The challenges of the project were doing the player character's parts and designing how the destructible terrain tiles and trash block system should work.

Using 3d did not have that much effect besides flybot's rotation animation. Another animation where 3d was slightly helpful was Robo's walking cycle, and even more so with the first walk cycle version which had sideways motion.

SOURCES

McKinley, M. 2006. The Game Animators Guide To Maya. Indianapolis, Indiana: Wiley Publishing.

Seegmiller, D. 2007. Digital Character Painting Using Photoshop CS3. Boston: Charles River Media.

Sims, A. & Isner, M. 2004. Experience XSI 4: The Official SOFTIMAGE XSI 4 Guide to Character Creation. Course Technology.

Steed, P. 2005. Modeling a Character in 3DS Max 2nd Edition. Plano, Texas: Wordware Publishing.

Thomas, F. & Johnston, O. 1995. Illusion Of Life. Disney Editions.

Ward, A. 2005. Game Character Development with Maya. Berkeley: New Riders.

Ward, A. 2008. Game Character Development: Digital Sculpting for the Realtime Artist. Course Technology.

WWW

Angel, E. 2008. Characters in Games. Available: http://www.cs.unm.edu/~angel/GAME/ . (Read: 14.10.2011).

8-Bit-Rocket. 2008. The basics of tile sheet animation (or blitting). Available: http://www.retrohello.com/2008/7/2/Tutorial-AS3-The-basics-of-tile-sheet-animation-or-blitting/ (Read: 25.10.2011).

Borzel, T. 2008. Tileset Tutorial in English. Available http://mistraal.wordpress.com/2008/04/19/tileset-tutorial-english/ (Read 7.11.2011).

Softimage|XSI Version 5.1 Animation. 2006. Avid Technology.

Softimage|XSI Version 5.1 Character Animation. 2006. Avid Technology.

Softimage|XSI Version 5.1 Nonlinear Animation. 2006. Avid Technology.

Softimage|XSI Version 5.1 Rendering. 2006. Avid Technology.

LIST OF APPENDICES

<< August >>

Team calendar.
August, 2010

month today << 2010 >>

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
						1 30 Add Event
2 31 Add Event	3 31 Add Event	4 31 Add Event	5 31 Add Event	6 31 Add Event	7 31 Add Event	8 31 Add Event
9 32 Add Event	10 32 Add Event	11 32 Add Event	12 32 Add Event	13 32 Add Event	14 32 Add Event	15 32 Add Event
16 33 Add Event	17 33 Add Event	18 33 Add Event	19 33 Add Event	20 33 Add Event ▪ gfx: schedule	21 33 Add Event ▪ gfx: designing, sketching	22 33 Add Event ▪ gfx: designing, sketching
23 34 Add Event ▪ gfx: robo-placeholder models	24 34 Add Event ▪ gfx: robo-rigging	25 34 Add Event ▪ gfx: anims-robo	26 34 Add Event ▪ gfx: robo spritesheets	27 34 Add Event ▪ gfx: desert placeholder graphics	28 34 Add Event ▪ gfx: designing, sketching	29 34 Add Event ▪ gfx: designing, sketching
30 35 Add Event ▪ gfx: UI placeholder	31 35 Add Event ▪ gfx: enemy placeholders					

v3.8.4 (8/18/2009) about... event styles

Category: Schedules

<< September >>

Team calendar.

September, 2010

month today << 2010 >>

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
		1 35 Add Event ▪ gxf: fx-placeholders (explosions, bullets etc.)	2 35 Add Event ▪ gfx: trash block designs	3 35 Add Event ▪ gfx: Sarai concept art	4 35 Add Event	5 35 Add Event
6 36 Add Event ▪ sk: Sarai sketches	7 36 Add Event ▪ sk: enemy bots	8 36 Add Event ▪ gfx: drops (enemy drops) ▪ gfx: guardB - sketching animations	9 36 Add Event ▪ gfx: 3d guard bot (modeling, rigging)	10 36 Add Event ▪ gfx: 3d guard bot (texturing?, spritesheets) ▪ gfx: toon shader tests?	11 36 Add Event	12 36 Add Event
13 37 Add Event ▪ gfx: flybot sketches	14 37 Add Event ▪ gfx: flybot modeling	15 37 Add Event ▪ gfx: flybot bomb, explosion ▪ gfx: robo textures	16 37 Add Event ▪ gfx: robo textures	17 37 Add Event ▪ gfx: robo toon shader sprites	18 37 Add Event	19 37 Add Event
20 38 Add Event	21 38 Add Event	22 38 Add Event	23 38 Add Event	24 38 Add Event	25 38 Add Event	26 38 Add Event
27 39 Add Event	28 39 Add Event	29 39 Add Event	30 39 Add Event			

v3.8.4 (8/18/2009) about...

event styles

Category: Schedules

<< November >> **Team calendar.** November, 2010 month today << 2010 >>

Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
1 44 Add Event	2 44 Add Event	3 44 Add Event	4 44 Add Event	5 44 Add Event	6 44 Add Event	7 44 Add Event
8 45 Add Event	9 45 Add Event	10 45 Add Event	11 45 Add Event	12 45 Add Event	13 45 Add Event	14 45 Add Event
15 46 Add Event	16 46 Add Event ▪ gfx: bg ▪ gfx: coil-gun fx	17 46 Add Event ▪ gfx: bg objs	18 46 Add Event	19 46 Add Event ▪ gfx: bg layers / bg objs	20 46 Add Event	21 46 Add Event
22 47 Add Event ▪ gfx: bg	23 47 Add Event ▪ gfx: bg / bg obj layers - scroll ▪ gfx: bg obj: big robot	24 47 Add Event ▪ SEMINAR: GAME DESIGN	25 47 Add Event ▪ gfx: desert top tiles, center tiles refine	26 47 Add Event ▪ gfx: sarai	27 47 Add Event	28 47 Add Event
29 48 Add Event	30 48 Add Event					

v3.8.4 (8/18/2009) about... event styles

Illustration 24. Robo's sprite sheet

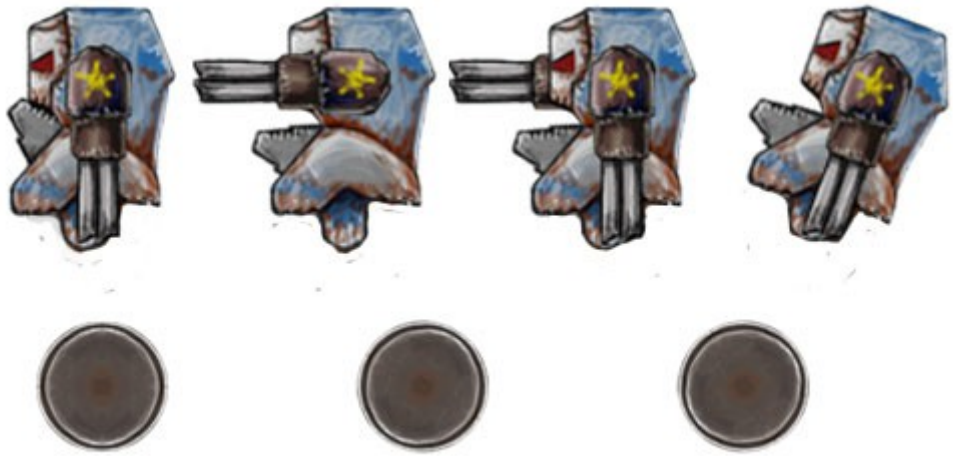


Illustration 25. Guardbot's sprite sheet

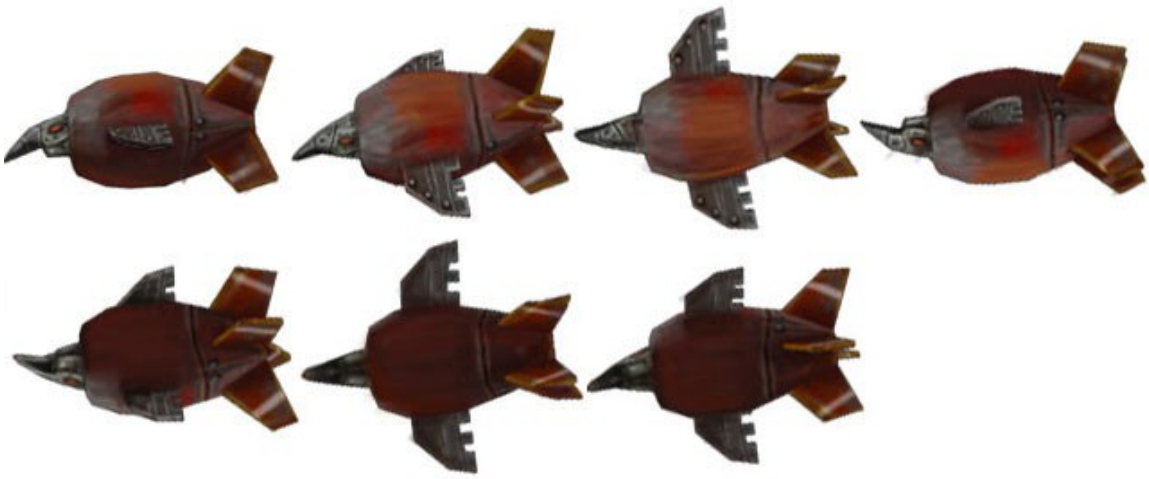


Illustration 26. Flybot's sprite sheet

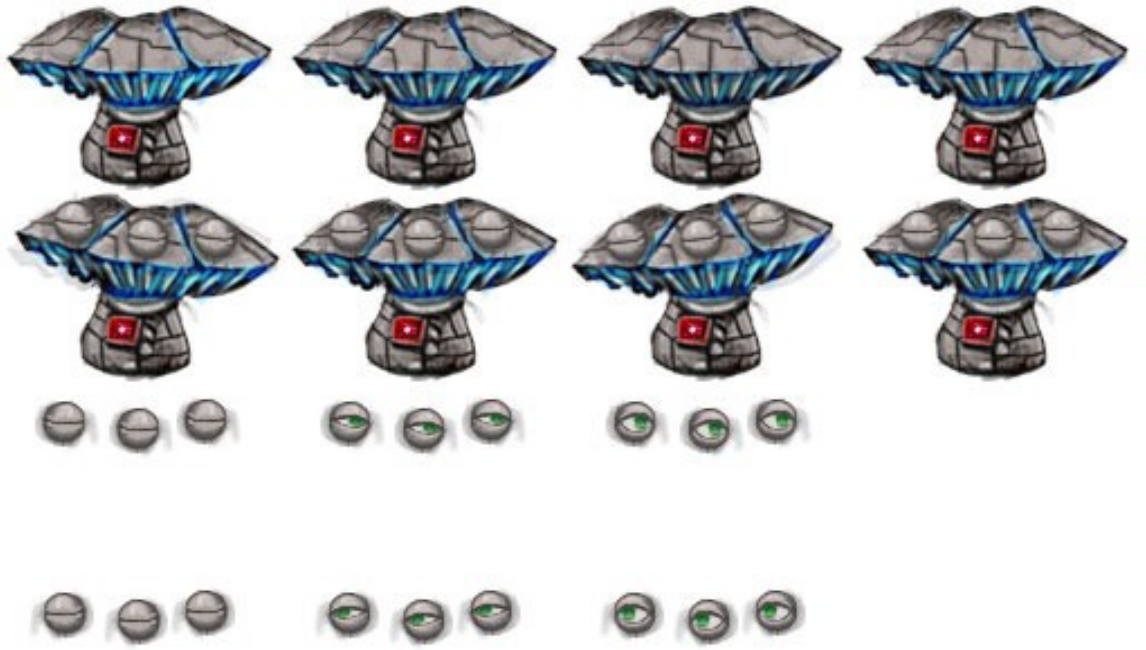


Illustration 27. Shroombot's sprite sheet

Illustration 28. Frobot's sprite sheet

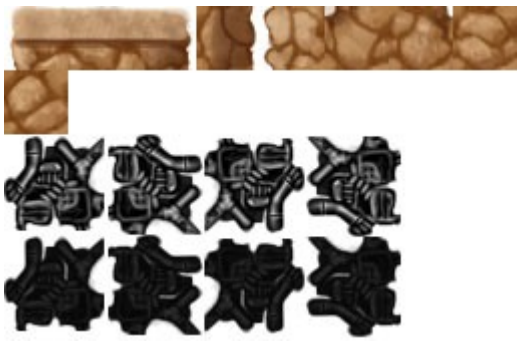


Illustration 29. Desert level tiles